To: T10 Technical Committee
From: Rob Elliott, HP (elliott@hp.com) and Jim Jones, Quantum (jim.jones@quantum.com)
Date: 2 July 2003
Subject: T10/03-186r1 SAS-1.1 Transport layer retries

Revision History
Revision 0 (6 May 2003) first revision
Revision 1 (2 July 2003) added more details

Related Documents
sas-r04a – Serial Attached SCSI revision 4a
03-229r0 SAS-1.1 Transport layer retries ladders

Overview
In SAS-1, errors transmitting frames result in either the logical unit terminating the command with CHECK CONDITION status, or the application client aborting the command. Possible errors are:

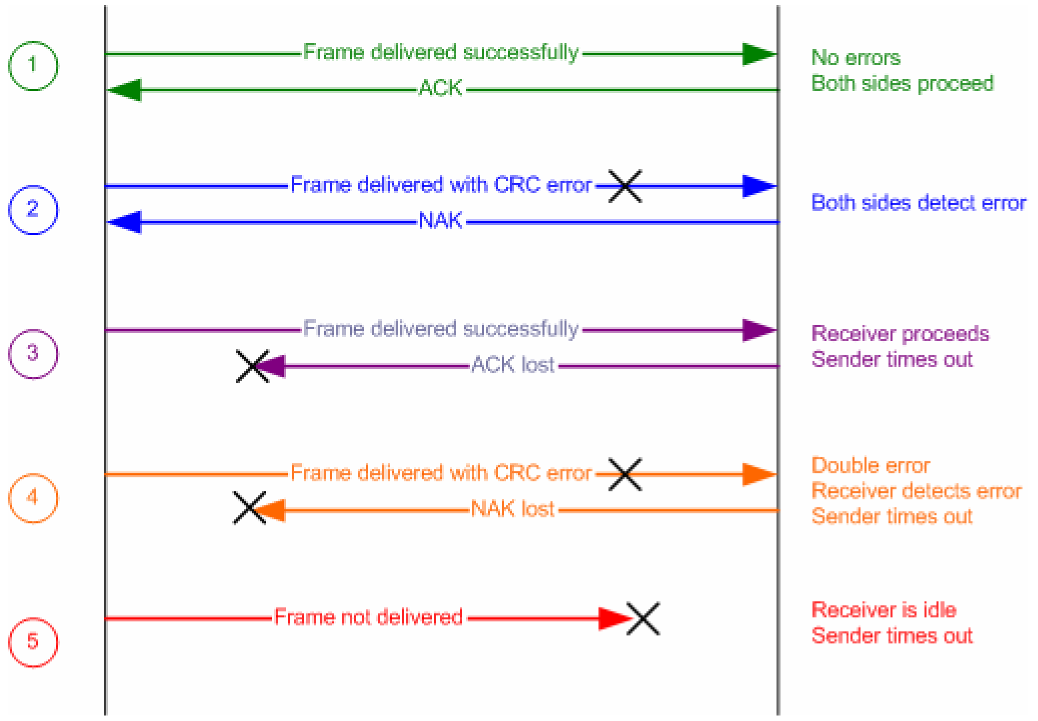

Figure 1. Error cases

Depending on the type of frame, different things happen in SAS-1.0:

Table 1. SAS-1.0 behavior

	COMMAND or TASK (I to T)	**XFER_RDY (T to I)**	**RESPONSE (T to I)**	**read DATA (T to I)**	**write DATA (I to T)**
1. Frame arrives OK; ACK arrives OK	Target runs the command after sending ACK.	Initiator replies with write DATA frame(s).	Both finish the command. Initiator can reuse tag after evidence of target progression.	Move on to more data or RESPONSE (or XFER_RDY for bidi commands)	Move on to more data, XFER_RDY, or RESPONSE (or read DATA for bidi commands)
2. Frame arrives with CRC error; NAK arrives OK	Target idle after sending NAK. Initiator sees the NAK and can resend the command.	Target terminates command.	Target can resend with retransmit=1.	Target terminates command	Initiator aborts command
3. Frame arrives OK; ACK lost	Target runs the command after sending ACK. I to T direction hung interlocked. Initiator detects ACK/NAK timeout in 1 ms. T to I direction could deliver DATA, XFER_RDY, or RESPONSE frames in that time. *Initiator cannot tell between ACK lost, NAK lost, and frame lost w/o using QUERY TASK.*	T to I hung interlocked. Target detects ACK/NAK timeout in 1 ms. I to T direction could deliver write DATA frames in that time. *Target cannot tell between ACK lost, NAK lost, and frame lost.*	T to I hung interlocked. Target detects ACK/NAK timeout in 1 ms. I to T direction not supposed to deliver new command with the same tag yet. Target resends with retransmit=1 in a new connection.	Subsequent ACKs/NAKs for data in flight are misassigned by the target. Target ACK/NAK timeout. Target terminates cmd.	Subsequent ACK/NAKs for data in flight are misassigned by the initiator. Initiator ACK/NAK timeout. Initiator aborts cmd.
4. Frame arrives with CRC error; NAK lost (double error)	Target idle after sending NAK. I to T direction hung interlocked. Initiator detects ACK/NAK timeout in 1 ms. Initiator can resend cmd later. *Initiator cannot tell between ACK lost, NAK lost, and frame lost w/o using QUERY TASK.*	T to I hung interlocked. Target detects ACK/NAK timeout in 1 ms. *Target cannot tell between ACK lost, NAK lost, and frame lost.*	T to I hung interlocked. Target detects ACK/NAK timeout in 1 ms. Target resends with retransmit=1 in new connection.	Subsequent ACKs/NAKs for data in flight are misassigned by the target. Initiator sees data offset gap. Target ACK/NAK timeout. Target terminates command. Initiator aborts cmd because of gap (if subseq. DATA frames)	Subsequent ACK/NAKs for data in flight are misassigned by the initiator. Initiator ACK/NAK timeout. Initiator aborts cmd. Target terminates command because of gap (if subseq. DATA frames). Initiator Response Timeout may occur causing target to terminate cmd.
5. Frame lost	Target idle. I to T direction hung interlocked. Initiator detects ACK/NAK timeout in 1 ms. Initiator can resend cmd later. *Initiator cannot tell between ACK lost, NAK lost, and frame lost w/o using QUERY TASK.*	T to I hung interlocked. Target detects ACK/NAK timeout in 1 ms. Target terminates cmd. *Target cannot tell between ACK lost, NAK lost, and frame lost.*	T to I hung interlocked. Target detects ACK/NAK timeout in 1 ms. Target resends with retransmit=1 in new connection.	Subsequent ACK/NAKs for data in flight are misassigned by the target. Initiator sees data offset gap. Target ACK/NAK timeout. Target terminates command. Initiator aborts cmd because of gap.	Subsequent ACK/NAKs for data in flight are misassigned by the initiator. Initiator ACK/NAK timeout. Initiator aborts cmd. Target terminates command because of gap (if subseq. DATA frames). Initiator Response Timeout may occur causing target to terminate cmd.

Proposed enhancements

Some backup applications will fail the whole backup if any of their commands are terminated with CHECK CONDITION. A way to retransmit frames is desirable.

These special recovery features are not always wanted; per-logical unit controls are needed. Tape drives would implement these features; disk drives probably would not. A Mode page per LUN to enable/disable special recovery mode (not the Enable Modify Data Pointers bit in the Disconnect-Reconnect mode page) would be added.

RESPONSE problems

- If target gets a NAK, resend as in SAS-1.0 (with **Retransmit**=1). **Retransmit**=1 isn't strictly necessary since the target didn't see anything; but it helps logic analyzers.
- If target detects an ACK/NAK timeout, resend as in SAS-1.0 (with **Retransmit**=1)

COMMAND or TASK problems

- If initiator gets a NAK, resend as in SAS-1.0
- If initiator detects a RESPONSE frame before it gets an ACK for a command, an ACK was lost. Treat the command as successful. Still close the connection with ACK/NAK Timeout, but do nothing more.
- If initiator detects an XFER_RDY or read DATA frame before it gets an ACK for a command, an ACK was lost. Treat the command as successfully received. Still close the connection with ACK/NAK Timeout, but start servicing the XFER_RDY or read DATA.
- If initiator detects an ACK/NAK timeout, close connection with ACK/NAK Timeout and send QUERY TASK in a new connection to see if the command is running in the target:
    - If FUNCTION SUCCEEDED, then assume it's running fine.
    - If FUNCTION COMPLETE, then resend the command.
    - Continue watching for a RESPONSE frame while doing this.
    - If a RESPONSE shows up, don't try to abort the command and don't report it as aborted to the application client.
    - if QUERY TASK is not supported, initiator will have to use ABORT TASK and try to abort the command (it may get a RESPONSE anyway). Logical units behind target ports supporting transport layer retries shall support QUERY TASK.

XFER_RDY problems

- If target gets a NAK, resend with **Retransmit**=1 and a new TPTT. **Retransmit**=1 isn't strictly necessary since the initiator didn't see anything, but it helps logic analyzers. New TPTT isn't strictly necessary since the old one isn't being used - however, this seems simpler
- If target detects an ACK/NAK timeout, resend with **Retransmit**=1 and a new TPTT
- If target sees write DATA frames before seeing an ACK for the XFER_RDY, target discards them
- Write DATA frames in response to an XFER_RDY with **Retransmit**=1 do not themselves have **Retransmit**=1
- Target shall change Target Port Transfer Tag between XFER_RDYs (either retransmitted or normal)
- Open issue: can we require the initiator stop sending for a previous XFER_RDY when a new one arrives for the same command (same tag)? This applies to both **Retransmit**=1 or **Retransmit**=0 cases. Proposal: initiator can continue sending data for the previous XFER_RDY, but must not send data for the new XFER_RDY until it is done sending for the previous. Once it sends a data frame for the new XFER_RDY, no more data frames for the old are allowed.
- When can target reuse the TPTT? This depends on when the initiator stops sending the data frames for the previous XFER_RDY. It would be easiest for the target if the initiator had to stop immediately; it might be easier for the initiator to let it continue until it receives a RESPONSE for the command. Proposal: The target can reuse the first TPTT when it sees a data frame with the second TPTT.

- If initiator receives an XFER_RDY with **Retransmit**=0 but it is already servicing an XFER_RDY, it should assume an error and abort the command. This is more likely a bug than a single bit error worth recovering from.

Write DATA problems

- Set **Retry Data Frames**=1 bit in XFER_RDY to tell the initiator to retry write DATA frames that encounter errors. Only target ports supporting transport layer retries would set this bit (and only if their mode page is thus enabled).
- If initiator gets a NAK, resend that write DATA frame and all that followed. It may go back to the last ACK/NAK balance point or the last XFER_RDY base offset. Going back to the frame in error rather than the ACK/NAK balance point is risky because a lost ACK or NAK could point to the wrong frame. Open issue: Going back to XFER_RDY base offset could be easier on the target and should probably be mandated. Proposal: XFER_RDY base offset only.
- If initiator detects an ACK/NAK timeout, resend all write DATA frames since the last ACK/NAK balance point or the last XFER_RDY base offset. Open issue: Going back to XFER_RDY base offset could be easier on the target and should probably be mandated. Proposal: XFER_RDY base offset only.
- Each resent write DATA frame has the correct Data Offset
- No use of the **Retransmit** bit for write DATA frames
- First resent write DATA frame has **Changing Data Pointer**=1
- If an ACK (rather than a NAK) was lost, the target doesn't know and may follow with a RESPONSE or XFER_RDY that surprises the initiator (in a subsequent connection)
    - If resent write DATA frames cross a RESPONSE frame, initiator has to accept the RESPONSE. The target discards subsequent write data frames since they have a now-unknown tag and TPTT. Easiest if initiator has to stop sending frames for the command after receiving a RESPONSE. Proposal: Require an initiator accept the RESPONSE frame and complete the command without worrying about the missing ACK for one of its write DATA frames.
    - If resent write DATA frames cross an XFER_RDY, initiator has to accept the XFER_RDY. Target shall use a different Target Port Transfer Tag for the XFER_RDY to help differentiate them. Proposal: initiator has to stop sending write DATA frames for the old XFER_RDY before sending any write DATA frames for the new XFER_RDY..
- Receiving a RESPONSE frame or XFER_RDY does NOT serve as a link layer ACK for the outbound direction - an ACK/NAK timeout must still occur. The inbound frame must be ACKed and honored, though (the T to I direction does not timeout).
- Target shall not abort a command when it sees a relative offset error. Target shall start discarding data after the gap until it gets a frame with **Changing Data Pointer**=1 (unlike the initiator on read data gaps, target does not have the option of storing bad data in the wrong place). Target may discard resent data up to the gap if it chooses (rather than overwriting).

Read DATA problems

- If target gets a NAK, it shall resend that read DATA frame and all that followed. It shall go back to the last ACK/NAK balance point rather than retry starting from the read DATA frame it thinks was NAKed, because a lost ACK or NAK while pipelining could mean it mismatched the NAK.
- If target detects an ACK/NAK timeout, it shall resend all read DATA frames since the last ACK/NAK balance point.
- If target gets a NAK, it should stop sending read DATA frames.
- No use of **Retransmit** bit for read DATA frames
- Set the **Changing Data Pointer** bit to 1 when the target is going back to an earlier data offset
- Target frames shall be monotonically increasing after **Changing Data Pointer**=1
- Target must get ACK/NAK balance before throwing away data that might have to be retransmitted
- Initiator shall not abort a command when it sees a relative offset error. Initiator may start discarding data after the gap until it gets a frame with **Changing Data Pointer**=1. Initiator may discard resent data up to the gap if it chooses (rather than overwriting).

New bits for target ports supporting transport layer retries

Keep the **Retransmit** bit in the SSP frame header (byte 10 bit 1).

- SAS-1.0 only allowed this bit be used for RESPONSE frames.
- SAS-1.1 also allow it for XFER_RDY frames

New **Changing Data Pointer** bit for DATA frames (both read DATA and write DATA frames) (byte 10 bit 0)

- 1 = the data offset is being set to a non-monotonically increasing value. (it must always go backwards)

New **Retry Data Frames** bit for XFER_RDY frames only (byte 9 bit 2)

- The initiator has permission to retry write DATA frames that encounter NAKs or ACK/NAK timeouts

Add the Protocol-Specific Logical Unit mode page (18h) (new to SAS):

- **Optimized Recovery** bit (located anywhere)
- 1 = the device server in the logical unit instructs the target port to:
    - set XFER_RDY frame header **Retry Data Frames** bit to 1
    - retry read DATA frames that fail
    - accept write DATA frames with **Changing Data Pointer** = 1

Transition

If the target sends an XFER_RDY with **Retry Data Frames** set to 1 to an initiator that doesn't understand it:

- If the initiator checks reserved fields, it will abort the command. If it determines it is aborting every one of its write commands, it might be prudent to ignore the reserved field. If the initiator is SAS-1.1 cognizant but does not support the feature, SAS-1.1 will require it to ignore the field and investigate why the mode page is set wrong.
- If the initiator does not check reserved fields, it will ignore it and never try to retry. If it encounters an ACK/NAK timeout or a NAK, it will abort the command.

If the initiator sends **Changing Data Pointer**=1 to a SAS-1 target:

- If the target checks reserved fields, it will abort the command. This should only happen after an error, so that's how SAS-1 behaved anyway.
- If the target does not check reserved fields, it will ignore it, detect a relative offset error, and abort the command. This should only happen after an error, so that's how SAS-1 behaved anyway.

If the target sends **Changing Data Pointer**=1 to a SAS-1 initiator:

- If the initiator checks reserved fields, it will abort the command. This should only happen after an error, so that's how SAS-1 behaved anyway.
- If the initiator does not check reserved fields, it will ignore it, detect a relative offset error, and abort the command. This should only happen after an error, so that's how SAS-1 behaved anyway.